#10 AF 2124 $
10/8/03 2100

PATENT
Attorney Docket Nº AMI 99-0006 (P1642US00)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Havemose
Serial Nº	:	09/542,714
Filed	:	April 4, 2000
Group Art Unit	:	2124
Examiner	:	Tuan A. Vu
For	:	DYNAMIC COUPLING

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

CERTIFICATE OF MAILING 37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service on September 8, 2003, in a First Class envelope, with sufficient postage thereon, addressed to: MS Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

ReNea D. Berggren

ReNea D. Berggren DATED: September 8, 2003

Please find enclosed herewith three (3) copies of Appellants' Brief on Appeal.

Please charge the fee of **$320.00** for filing an Appeal Brief to Deposit Account Nº 50-0439. In the event that the Commissioner determines that any additional fees are required, or that any overpayment has been made, for this or any other Paper in this application, the Commissioner is hereby authorized to charge any such additional fees and to credit any overpayment to Deposit Account Nº 50-0439. A duplicate copy of this *Petition* is enclosed for accounting purposes only.

Please direct all correspondence to: **CUSTOMER NO. 32718**

MARK WALKER, ESQ.
GATEWAY, INC.
MS SD-21
14303 GATEWAY PLACE
POWAY, CA 92064
(858) 848-3449 TELEPHONE
(858) 848-2671 FACSIMILE

DATED: September 8, 2003.

Respectfully submitted,

Gateway, Inc.,

By Walter J. Malinowski
Walter J. Malinowski
Reg. Nº 43,423

ATTACHMENTS:

1. A copy of this *Transmittal* for accounting purposes.
2. Three (3) copies of *Appellant's Brief on Appeal.*
3. Return postcards

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Havemose	
Serial No.:	09/542,714	Art Unit: 2124
Filed:	April 4, 2000	Examiner: Tuan A. Vu
For:	*DYNAMIC COMPILING*	

Mail Stop Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF ON APPEAL

This is an appeal from the Final Office Action dated May 22, 2003, finally rejecting claims 1-44.

(1) REAL PARTY IN INTEREST

The real party in interest is Gateway, Inc.

(2) RELATED APPEALS AND INTERFERENCES

Appellant is not aware of any related appeals or interferences.

(3) STATUS OF CLAIMS

The status of the claims is as follows:

Claims allowed: none

Claims objected to: none

Claims rejected: Claims 1-44.

(4) STATUS OF AMENDMENTS AFTER FINAL

There has been one amendment proffered after the Final Office Action of May 22, 2003. This amendment, mailed July 21, 2003, made a typographical correction to the specification, made no changes to the claims, and has been entered by the Patent Office.

(5) SUMMARY OF INVENTION

The present invention provides an execution environment for optimizing the efficiency of the distributed object system. In accordance with the present invention, a compiler with ability to interpret, a just in time compiler, and a pre-compiler may be implemented together. A dynamic base object is provided in a tagged file format that allows a compiler to identify critical sections of the object code for immediate one-time compilations while delaying compilation of non-tagged, non-critical code until required by the system.

(6) ISSUES

I. Whether the Patent Office properly rejected Claims 1-5, 7-11, 13-16, 18, and 20-44 under 35 USC 103(a) as being unpatentable over Blandy, U.S. Patent No. 6,295,642, in view of Coutts et al., U.S. Patent No. 6,311,165?

II. Whether the Patent Office properly rejected Claims 6, 12, 17, and 19 under 35 U.S.C. 103(a) as being unpatentable over Blandy, U.S. Patent No. 6,295,642, and Coutts, et al., U.S. Patent No. 6,311,165, as applied to Claims 1, 7, 13, 18, and further in view of Hamby et al., U.S. Patent No. 5,848,274?

(7) GROUPING OF CLAIMS

For each ground of rejection that appellant contests herein which applies to more than one claim, such additional claims, to the extent separately identified and argued below, do not stand and fall together.

The Claims are at least as distinguishable as grouped below:

Group I: Claims 13, 17, 35 stand and fall together.

Group II: Claims 14-16 stand and fall together.

Group III: Claim 18 stands and falls alone.

Group IV: Claim 20 stands and falls alone.

Group V: Claims 1, 3-5, 7, 9-11, 21-23, 27-29, 31-33, 37-39, 42, and 43 stand and fall together.

Group VI: Claims 2 and 8 stand and fall together.

Group VII: Claims 24, 26, 36, and 40 stand and fall together.

Group VIII: Claims 25, 30, 34, and 41 stand and fall together.

Group IX: Claim 44 stands and falls alone.

Group X: Claim 17 stands and falls alone.

Group XI: Claims 6, 12, and 19 stand and fall together.

(8) ARGUMENT

ISSUE I

The first issue is whether the Patent Office properly rejected claims 1-5, 7-11, 13-16, 18, and 20-44 under 35 USC 103(a) as being unpatentable over Blandy, U.S. Patent No. 6,295,642, in view of Coutts et al., U.S. Patent No. 6,311,165.

GROUP I

The Patent Office rejected claims 1-5, 7-11, 13-16, 18, and 20-44 under 35 USC 103(a) as being unpatentable over Blandy, U.S. Patent No. 6,295,642, in view of Coutts et al., U.S. Patent No. 6,311,165.

When applying 35 U.S.C. 103, the following tenets of patent law must be adhered to: (A) the claimed invention must be considered as a whole; (B) the references must be considered as a whole and must suggest the desirability and thus the obviousness of making the combination; (C) the references must be viewed without the benefit of impermissible hindsight vision afforded by the claimed invention; and (D) reasonable expectation of success is the standard with which obviousness is determined. See MPEP § 2141 and Hodosh v. Block Drug Co., Inc., 786 F.2d 1136, 1143 n.5, 220 USPQ 182, 187 n.5 (Fed. Cir. 1986).

The Patent Office has based the rejection of all claims on a combination of Blandy and Coutts. In particular, the Patent Office has asserted that Blandy is deficient in not teaching "the independent byte-code includes at least one dynamic base object, the at least one dynamic base

object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." However, Claims 13-17 and 35 do not recite this limitation. That is, were Coutts to provide a valid teaching of this limitation, Coutts would not be needed because Claims 13-17 and 35 lack the limitation Coutts has been alleged to teach. Claims 13-17 and 35 recite "a loader ... being capable of ... pre-compiling," "an identifier coupled to the loader, the identifier suitable for identifying the tagged section of the byte-code," and "the identified tagged section is compiled by the compiler when the byte-code is loaded so as to enable the tagged section of byte-code to be utilized without additional compiling of the tagged section of byte-code." Addressing the first comment on the continuation sheet of the Advisory Action mailed August 12, 2003, the combination of these three limitations is not taught or suggested by Blandy. Blandy discloses a Just in Time station 224, a compiler 210, and an interpreter 212. Brandy teaches Just in Time compiling and interpreting, but not pre-compiling and not that "the identified tagged section is compiled by the compiler when the byte-code is loaded." The claims recited that the tagged section is identified by an identifier. Blandy, column 4, line 54, through column 5, line 23, discloses a compiler lock 227 which is used to serialize compilation of a method, an invoker field 228 of method block 226 that is set to point to the Just In Time initialization code, and branch monitors that may be implemented using breakpoints with interrupt handlers (i.e., through the operation of assembler language conditional jump instructions, as shown in column 5, lines 26-29). No identifier that identifies a tagged section of bytecode that is compiled when loaded is disclosed or suggested by Blandy. Thus, Blandy does not anticipate or make obvious Claims 13-17 and 35. Moreover, even if Blandy were modifiable by Coutts, Coutts does not teach or suggest the combination of these limitations and does not remedy the deficiencies of Blandy. Thus, Claims 13-17 and 35 are allowable over the prior art of record.

GROUP II

Claims 14-16 recite "an encoder for encoding the source code to byte-code" and "a tagger for tagging a section of the byte-code." Group II is separately patentable from Group I because of this limitation. Addressing the third comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Blandy does not teach or suggest a tagger for tagging a section of the byte-code. Figure 6 of Blandy does not disclose encoding; instead, Figure 6 illustrates a

process for a fall through monitor. Blandy, column 5, lines 14-23, disclose breakpoints corresponding to assembler language jump instructions. The jump instructions of Blandy refer to byte-code and not to a tagger that tags a section of the byte-code. Thus, Blandy does not teach or disclose a tagger or an encoder. Even if Blandy were modifiable by Coutts, Coutts does not teach or suggest the combination of these limitations. Thus, Claims 14-16 are allowable over the prior art of record.

GROUP III

Claim 18 recites "a method for providing an execution environment in an information appliance network, comprising: a) encoding an application source code in a processor independent byte-code; b) tagging at least some portion of said processor independent byte-code; and c) compiling at least some portion of said tagged processor independent byte-code, wherein the independent byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." Group III is separately patentable from Groups I and II because of this limitation. In the present application, on page 9, lines 23-26, interface and implementation DBOs are described such that "when an application creates a DBO, two DBOs are actually created (FIGS. 6A and 6B). These two DBOs are an interface-DBO within the application, and an instance of the real DBO (a/k/a an implementation-DBO)." On page 10, lines 3-5, the present specification further describes interface and implementation DBOs "In a preferred embodiment of the invention, each time the application uses the interface-DBO, a message is sent to the implementation-DBO, which carries out the task and returns the result. When the application frees the DBO the reverse happens." Blandy, as recognized by the Patent Office, does not teach "at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." Blandy shows a block diagram of a data processing system in Figure 1. Blandy shows a Java virtual machine in Figure 2. Blandy (column 1, lines 50-61) discloses Java bytecode may be stored as a Java application or servlet, (column 2, lines 1-5) refers to Just in Timing compiling, and (column 3, lines 45-54) discloses data processing system 100 may be a standalone system. None of the figures or sections of Blandy cited by the Patent Office relate to a dynamic base object. Coutts (column 25, line 66,

through column 26, line 10) discloses that Java byte codes may be executed directly in silicon. Coutts also discloses that using interpreters yields poor performance and Just In Time compilers require increased amounts of memory. Coutts shows a functional block diagram of a thin client application architecture in Figure 32. Coutts discloses (column 27, lines 26-50) a thin client used in point of sale applications. (Also, see the discussion of claim 4 regarding tagging.)

The Final Office Action, on pages 4 and 5, furthermore asserts (and the second comment of the Advisory Action mailed August 12, 2003, reasserts) that Blandy and Coutts suggest "wherein the independent byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." However, neither Blandy nor Coutts disclose or suggest this limitation. Applicant shows the relationship between the implementation DBO and interface DBO in Figures 4 and 6A of Applicant's application for patent (as reproduced below). There is no disclosure or suggestion in Blandy suggesting a need or desire for a "dynamic base object that includes an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under section 103, teachings of references can be combined only if there is some suggestion or incentive to do so. *ACS Hosp. Sys., Inc. v. Montefiore Hosp.,* 732 F.2d 1572, 221 USPQ 929 (Fed. Cir. 1984). The Examiner may not use the patent application as a basis for the motivation to combine or modify the prior art to arrive at the claimed invention. Thus, Claim 18 is allowable over the prior art of record.

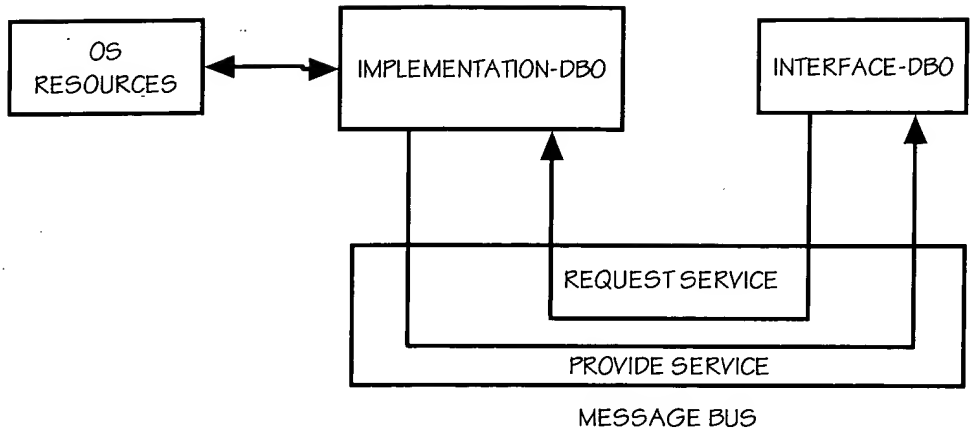
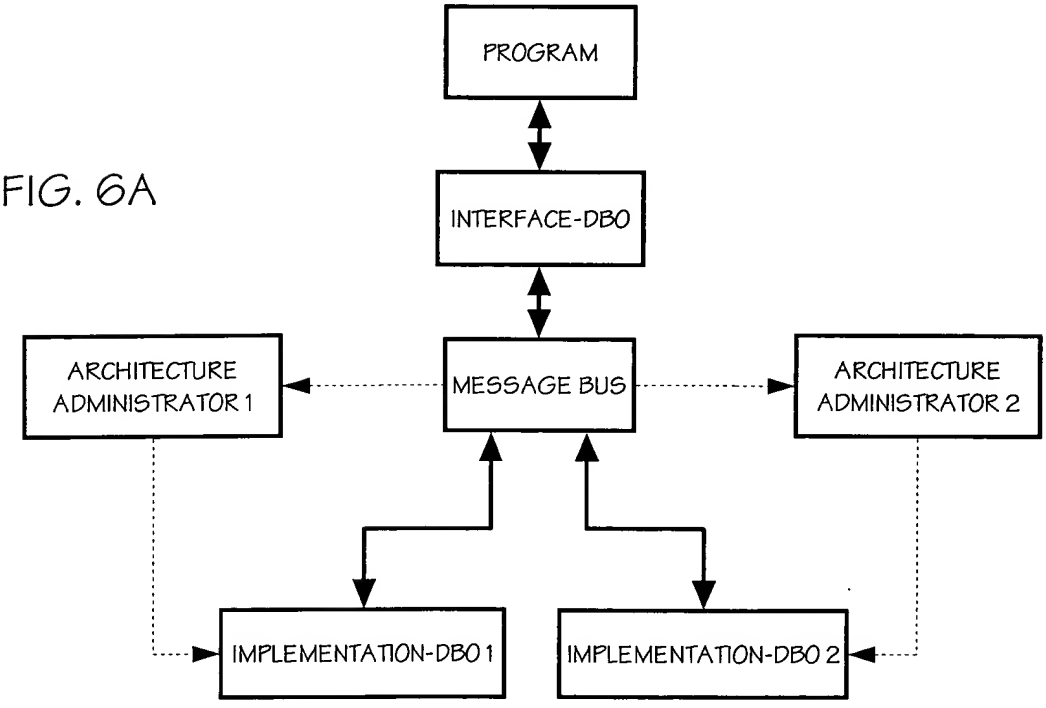

OS RESOURCES
IMPLEMENTATION-DBO
INTERFACE-DBO
REQUEST SERVICE
PROVIDE SERVICE
MESSAGE BUS

FIG. 4

FIG. 6A


PROGRAM
INTERFACE-DBO
ARCHITECTURE ADMINISTRATOR 1
MESSAGE BUS
ARCHITECTURE ADMINISTRATOR 2
IMPLEMENTATION-DBO 1
IMPLEMENTATION-DBO 2

GROUP IV

Claim 20 recites "compiling includes identifying the portion of said tagged processor independent byte-code." Group IV is separately patentable from Groups I-III because of this limitation. Blandy, column 4, line 54, through column 5, line 23, discloses a compiler lock 227 which is used to serialize compilation of a method, an invoker field 228 of method block 226 that is set to point to the Just In Time initialization code, and branch monitors that may be implemented using breakpoints with interrupt handlers (i.e., through the operation of assembler language conditional jump instructions, as shown in column 5, lines 26-29). No identifier that identifies a tagged section of bytecode that is compiled when loaded is disclosed or suggested by Blandy. Coutts does not teach or suggest these limitations. As the combination of Blandy and Coutts does not make obvious Claim 20, Claim 20 is allowable over the prior art of record for this reason as well as being dependent upon allowable Claim 18.

GROUP V

Claim 1-12, 21-34, and 36-44 recite "a method for dynamic compiling" that includes loading byte-code on a digital information appliance, said byte-code suitable for including a tagged section," "identifying the tagged section of the byte-code," "wherein the tagged section is compiled when the byte-code is loaded so as to enable the digital information appliance to utilize the tagged section of byte-code without additional compiling of the tagged section of byte-code by the digital information appliance," and "wherein the byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus." Group V is separately patentable from Groups I-IV because of this limitation. Contrary to the assertion of the third comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Blandy does not teach compiling the tagged section when the byte-code is loaded. The Patent Office cited column 3, lines 24-28, and column 4, lines 7-16, of Blandy as teaching this limitation. The first cited portion of Blandy on column 3, lines 24-28, asserts "Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive 126, and may be loaded into main memory 104 for execution by processor 102" and does not teach or suggest that the tagged

section is compiled when loaded. Any tagged section of the byte-code could be compiled before loading or after loading. The second cited portion of Blandy on column 4, lines 7-16, discloses a Just in Time compiler 210, an interpreter 212 and bytecodes described as a sequence of instructions, does not teach or suggest that the tagged section is compiled when loaded. Just In Time compiling refers to compilation immediately before execution. An interpreter translates and runs code at the same time. Coutts does not remedy the deficiencies of Blandy. Thus, Claims 1-12, 21-34, and 36-44 are not made obvious by Blandy in view of Coutts. Furthermore, as Claims 1-12, 21-34, and 36-44 have limitations found in both Claims 13-17 and 18-20, the combination of the limitations of these claims is believed to be novel and non-obvious and the above discussion of these claims applies (including pages 4 and 5 of the final office action and the second comment of the Advisory Action). Therefore, Claims 1-12, 21-34, and 36-44 are allowable over the prior art of record.

GROUP VI

Claims 2 and 8 recite "encoding application source code to byte-code, the byte-code including code in a processor-independent form which is suitable for further analysis and tagging a section of the byte-code." Group VI is separately patentable from Groups I to V because of this limitation. Figure 13 (see below) illustrates the process recited in claims 2 and 8. Blandy discloses (Figure 4, column 5, lines 14-23) jump points and breakpoints already found in the byte code and does not teach encoding the application code to byte-code and tagging a section of the byte-code. Blandy does not teach or suggest the limitations of Claims 2 or 8. Coutts is directed to a transaction processing system that uses a Java Virtual Machine built in silicon rather than use a Just In Time compiler or interpreter. Thus, Claims 2 and 8 are allowable over the prior art of record for this additional reason. (Regarding the fourth comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Applicant requests an elaboration by the Patent Office.)

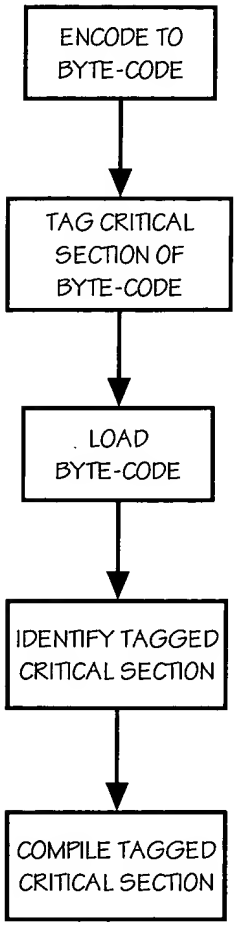


FIG. 13

GROUP VII

Claims 24, 26, 36, and 40 recite "wherein the dynamic base object is programmed through a scripting language with run-time object invocation." Group VII is separately patentable from Groups I to VI because of this limitation. Coutts, as discussed above does not disclose a dynamic base object. Although Coutts (column 11, lines 44-49) does disclose reports in HTML form, Coutts does not disclose a scripting language for programming a dynamic base object. As discussed regarding the rejection of Claim 18, neither Blandy nor Coutts teach or suggest "at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus," and

so do not teach or suggest a dynamic base object programmed through a scripting language with run-time object invocation." Thus, Claims 24, 26, 36, and 40 are allowable over the prior art of record for this additional reason. (Regarding the fourth comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Applicant requests an elaboration by the Patent Office.)

GROUP VIII

Claims 25, 30, 34, and 41 recite "the interface dynamic base object and the implementation dynamic base object communicate bi-directionally." Group VIII is separately patentable from Groups I to VII because of this limitation. Coutts (Figures 14-16, 30, 32) show transaction networks and terminals, but does not show at least one dynamic base object that has an interface dynamic base object and an implementation dynamic base object that communicate bi-directionally. Thus, Claims 25, 30, 34, and 41 are allowable over the prior art of record for this additional reason. (Regarding the fourth comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Applicant requests an elaboration by the Patent Office.)

GROUP IX

Claim 44 recites "wherein the at least one dynamic base object is fully thread safe." That is, the at least one dynamic base object may be utilized by concurrent multiple thread applications. Group IX is separately patentable from Groups I to VIII because of this limitation. Coutts (column 4, lines 54-59) discloses the compiler lock prevents a method from being compiled by multithreads. Thus, Claim 44 is allowable over the prior art of record. (Regarding the fourth comment of the continuation sheet of the Advisory Action mailed August 12, 2003, Applicant requests an elaboration by the Patent Office.)

ISSUE II

The second issue is whether the Patent Office properly rejected Claims 6, 12, 17, and 19 under 35 U.S.C. 103(a) as being unpatentable over Blandy, U.S. Patent No. 6,295,642, and Coutts, et al., U.S. Patent No. 6,311,165, as applied to Claims 1, 7, 13, 18, and further in view of Hamby et al., U.S. Patent No. 5,848,274.

GROUP X

Claim 17 recites "the loader includes a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system." Group X is separately patentable from Groups I to IX because of this limitation in combination with other limitations. No basis has been provided for rejecting Claim 17 by the combination of Blandy and Coutts because Coutts has been cited by the Patent Office for a teaching about dynamic base objects – a limitation not present in Claim 17. Contrary to the assertion in the fifth comment of the Advisory Action mailed August 12, 2003, Blandy does not disclose a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system. Blandy, on column 4, lines 54-59, ensures that the same method is not compiled simultaneously by multiple threads and does not teach or suggest "a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system." Even if Blandy were modifiable by Coutts, Coutts does not teach or suggest a validation utility. Coutts, on column 44, lines 19-39, is directed to messages and not byte-code. Hamby, on column 27, line 42, through column 28, line 6, discloses an incremental byte compiler, but does not disclose "a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system." Interrogation of the operating system to find a file (col. 28, lines 7-8, Hamby) is not validation that the byte-code conforms with byte-code suitable for utilization by the system." The last six lines of page 11 and the first two lines of page 12 of the Final Office Action asserting byte-code validation is not a fair reading of Blandy, Coutts, and/or Hamby, and does not conform to the cited passages from those three references. Thus, Claim 17 is allowable over the prior art on its own merit as well as its dependency from allowable base claim 13.

GROUP XI

Claims 6, 12, and 19, similarly to Claim 17, recite validating the byte-code. Group XI is separately patentable from Groups I to X because of this limitation in combination with other limitations. Blandy does not disclose a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system. Blandy, on column 4, lines 54-59, ensures that the same method is not compiled simultaneously by multiple threads and does not teach or suggest "a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system." Even if Blandy were modifiable by Coutts, Coutts does not teach or

suggest a validation utility. Coutts, on column 44, lines 19-39, is directed to messages and not byte-code. Hamby, on column 27, line 42, through column 28, line 6, discloses an incremental byte compiler, but does not disclose "a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system." Interrogation of the operating system to find a file (col. 28, lines 7-8, Hamby) is not validation that the byte-code conforms with byte-code suitable for utilization by the system." The last six lines of page 11 and the first two lines of page 12 of the Final Office Action asserting byte-code validation is not a fair reading of Blandy, Coutts, and/or Hamby, and does not conform to the cited passages from those three references. Thus, Claims 6, 12, and 19 are allowable over the prior art on their own merit as well as their dependency from base claims 1, 7, and 18.

CONCLUSION

The sixth and final comment of the continuation sheet of the Advisory Action mailed August 12, 2003, asserts "if some arguments have some valid points, further search and reconsideration are needed." Section 707.07(j) of the MPEP at least implies a duty on the Patent Office to indicate allowable subject matter. Applicant requests clarification of the language "if some arguments have some valid points, further search and reconsideration are needed" because the amendment after final did not include any modification of the claims. The just noted language of the advisory action implies that there is allowable subject matter in the claims that has not been indicated by the Patent Office.

For the reasons provided above, it is respectfully requested that in each of the rejections discussed herein under 35 U.S.C. § 103, the Patent Office has failed to meet the burden in establishing a *prima facie* basis for the rejections of Claims 1-44. Accordingly, reversal of all outstanding rejections is earnestly solicited.

Respectfully submitted,

GATEWAY, INC.,

Dated: September 8, 2003

By: Walter J. Malinowski
Walter J. Malinowski
Reg. No. 43,423

Walter J. Malinowski
SUITER • WEST PC LLO
14301 FNB Parkway, Suite 220
Omaha, NE 68154
(402) 496-0300 telephone
(402) 496-0333 facsimile

(9) CLAIMS

1. A method for dynamic compiling, comprising:

loading byte-code on a digital information appliance, said byte-code suitable for including a tagged section;

identifying the tagged section of the byte-code; and

compiling the tagged section of byte-code;

wherein the tagged section is compiled when the byte-code is loaded so as to enable the digital information appliance to utilize the tagged section of byte-code without additional compiling of the tagged section of byte-code by the digital information appliance,

wherein the byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus.

2. The method as described in claim 1, further comprising:

encoding application source code to byte-code, the byte-code including code in a processor-independent form which is suitable for further analysis; and

tagging a section of the byte-code.

3. The method as described in claim 2, wherein the byte-code including code in the processor-independent form is suitable for further analysis including at least one of suitable for compiler optimization, suitable for processing by interpreters, and suitable for use in generation of binary instruction for the digital information appliance processing system.

4. The method as described in claim 2, wherein the section of the byte-code is tagged for being performance sensitive.

5. The method as described in claim 1, further comprising storing the compiled tagged section of byte-code in persistent storage.

6. The method as described in claim 1, wherein loading includes validating that the byte-code conforms with byte-code suitable for utilization by the digital information appliance.

7. A digital information appliance suitable for dynamic coupling, comprising:

a processor for implementing a program of instructions; and

a memory for storing the program of instructions, the program of instructions suitable for configuring the digital information appliance to load byte-code, said byte-code suitable for including a tagged section;

identify the tagged section of the byte-code; and

compile the tagged section of byte-code;

wherein the tagged section is compiled when the byte-code is loaded so as to enable the digital information appliance to utilize the tagged section of byte-code without additional compiling of the tagged section of byte-code by the digital information appliance,

wherein the byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus.

8. The digital information appliance as described in claim 7, further comprising:

encoding application source code to byte-code, the byte-code including code in a processor-independent form which is suitable for further analysis; and

tagging a section of the byte-code.

9. The digital information appliance as described in claim 8, wherein the byte-code including code in the processor-independent form is suitable for further analysis including at least one of suitable for compiler optimization, suitable for processing by interpreters, and suitable for use in generation of binary instruction for a digital information appliance's processing system.

10. The digital information appliance as described in claim 7, wherein the section of the byte-code is tagged for being performance sensitive.

11. The digital information appliance as described in claim 7, further comprising storing the compiled tagged section of byte-code in persistent storage.

12. The digital information appliance as described in claim 7, wherein loading includes validating that the byte-code conforms with byte-code suitable for utilization by the digital information appliance.

13. A system for providing an execution environment that is suitable for dynamic compiling, comprising:

a memory device suitable for storing computer readable information;

a loader coupled to the memory device, the loader suitable for loading byte-code to the memory, said byte-code suitable for including a tagged section, the loader being capable of interpreting, just in time compiling, and pre-compiling;

an identifier coupled to the loader, the identifier suitable for identifying the tagged section of the byte-code;

a compiler coupled to the identifier;

wherein the identified tagged section is compiled by the compiler when the byte-code is loaded so as to enable the tagged section of byte-code to be utilized without additional compiling of the tagged section of byte-code.

14. The system as described in claim 13, further comprising:

an encoder for encoding application source code to byte-code, the byte-code including code in a processor-independent form which is suitable for further analysis; and

a tagger for tagging a section of the byte-code.

15. The system as described in claim 14, wherein the byte-code including code in the processor-independent form is suitable for further analysis including at least one of suitable for compiler optimization, suitable for processing by an interpreter, and suitable for use in generation of binary instruction for a processing system.

16. The system as described in claim 14, wherein the section of the byte-code is tagged for being performance sensitive.

17. The system as described in claim 13, wherein the loader includes a validation utility for validating the byte-code conforms with byte-code suitable for utilization by the system.

18. A method for providing an execution environment in an information appliance network, comprising:

a) encoding an application source code in a processor independent byte-code;

b) tagging at least some portion of said processor independent byte-code; and

c) compiling at least some portion of said tagged processor independent byte-code,

wherein the independent byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus.

19. The execution environment for an information appliance network of claim 18, further including validating at least some portion of said processor independent byte-code.

20. The execution environment for an information appliance network of claim 18, wherein compiling includes identifying the portion of said tagged processor independent byte-code.

21. The method of Claim 1, wherein the message bus provides interprocessor communications within a system.

22. The method of Claim 1, wherein the message bus provides communications over the network.

23. The method of Claim 1, wherein the byte-code is processor independent.

24. The method of Claim 1, wherein the dynamic base object is programmed through a scripting language with run-time object invocation.

25. The method of Claim 1, wherein the interface dynamic base object and the implementation dynamic base object communicate bi-directionally.

26. The method of Claim 7, wherein the message bus provides interprocessor communications within a system.

27. The method of Claim 7, wherein the message bus provides communications over the network.

28. The method of Claim 7, wherein the byte-code is processor independent.

29. The method of Claim 7, wherein the dynamic base object is programmed through a scripting language with run-time object invocation.

30. The method of Claim 7, wherein the interface dynamic base object and the implementation dynamic base object communicate bi-directionally.

31. The system of Claim 13, wherein the byte-code includes at least one dynamic base object, the at least one dynamic base object comprising an interface dynamic base object and an implementation dynamic base object that communicate with each other over a message bus.

32. The system of Claim 31, wherein the message bus provides interprocessor communications within a system.

33. The system of Claim 31, wherein the message bus provides communications over the network.

34. The system of Claim 31, wherein the interface dynamic base object and the implementation dynamic base object communicate bi-directionally.

35. The system of Claim 13, wherein the byte-code is processor independent.

36. The system of Claim 31, wherein the dynamic base object is programmed through a scripting language with run-time object invocation.

37. The method of Claim 18, wherein the message bus provides interprocessor communications within a system.

38. The method of Claim 18, wherein the message bus provides communications over the network.

39. The method of Claim 18, wherein the independent byte-code is processor independent.

40. The method of Claim 18, wherein the dynamic base object is programmed through a scripting language with run-time object invocation.

41. The method of Claim 18, wherein the interface dynamic base object and the implementation dynamic base object communicate bi-directionally.

42. The method of Claim 1, wherein the interface dynamic base object includes multiple lines of code.

43. The method of Claim 1, wherein the digital information appliance is a thin network appliance.

44. The method of Claim 1, wherein the at least one dynamic base object is fully thread safe.